

Web Programming with PHP

Christophe Lecoutre
lecoutre@cril.fr

IUT de Lens - CRIL CNRS UMR 8188
Université d'Artois
France

Département SRC - 2011/2012

Outline

- 1 Introduction
- 2 Principes de base
- 3 Les structures de contrôle
- 4 Les chaînes de caractères
- 5 Les tableaux
- 6 Les fonctions
- 7 Modules et fonctions prédéfinis
- 8 Les formulaires
- 9 Cookies et Sessions
- 10 Accès à MySQL

- 1 Introduction
- 2 Principes de base
- 3 Les structures de contrôle
- 4 Les chaînes de caractères
- 5 Les tableaux
- 6 Les fonctions
- 7 Modules et fonctions prédéfinis
- 8 Les formulaires
- 9 Cookies et Sessions
- 10 Accès à MySQL

Books

- Jean Engels
PHP5,
Eyrolles.
- ...



Sites

- Le site officiel sur <http://www.php.net>
 - Le Manuel en anglais sur <http://php.net/manual/en>
 - Le Manuel en français sur <http://fr.php.net/manual/fr>
-
- PHP-index sur <http://www.phpindex.com>
 - PHP-scripts sur <http://www.phpscripts-fr.net>
 - PHP sur dmoz à <http://www.dmoz.org>

Description Générale

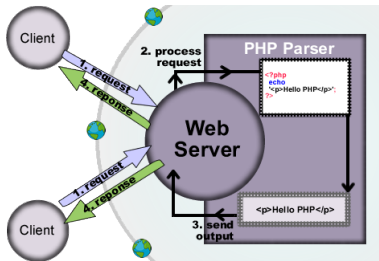
PHP (Hypertext Preprocesso) est un langage de scripts principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP. Il sert à produire et renvoyer aux navigateurs des documents HTML construit par le moteur de script Zend Engine 2.

Le langage PHP est utilisé principalement en tant que langage de script côté serveur, ce qui veut dire que c'est le serveur (la machine qui héberge la page Web en question) qui va interpréter le code PHP et générer du code (constitué généralement d'XHTML/HTML, de CSS, et parfois de JavaScript) qui pourra être interprété par un navigateur.

Quelques autres langages de scripts côté serveur :

- ASP (Active Server Pages)
- JSP (Java Server Pages)
- Perl
- Python
- Ruby

Description Générale



Système LAMP :

- Linux
- Apache
- MySQL
- PHP

Est-ce que votre serveur web local est opérationnel ?

http://localhost/

Quels modules de PHP sont installés sur votre serveur (local ou distant) ?

- 1 Créer le fichier `info.php` suivant :

```
<?php
    phpinfo();
?>
```

- 2 transférer/copier-le à la racine de votre site que l'on appellera `leSite`
- 3 saisissez l'adresse `http://leSite/info.php`

Remarque


La page que vous obtenez (voir diapositive suivante) vous donne un certain nombre d'informations utiles.

phpinfo() - Mozilla Firefox

File Edit View History Bookmarks Tools Help

phpinfo()

PHP Version 5.3.2-1ubuntu4.2




System	Linux arvo 2.6.32-21-generic #32-Ubuntu SMP Fri Apr 16 08:10:02 UTC 2010 #686
Build Date	May 13 2010 19:49:13
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/gd.ini, /etc/php5/apache2/conf.d/mcrypt.ini, /etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d/pdo_mysql.ini
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626.NTS
PHP Extension Build	API20090626.NTS
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

This server is protected with the Suhosin Patch 0.9.9.1
Copyright (c) 2006-2007 [Hardened-PHP Project](#) Copyright (c) 2007-2009
SektionEins GmbH

수호신

This program makes use of the Zend Scripting Language Engine:
Zend Engine v2.3.0, Copyright (c) 1998-2010 Zend Technologies

Powered By


Done

- 1 Créer le fichier `leSite.conf` sous `/etc/apache2/sites-enabled` tel que :

```
<VirtualHost *:80>
  ServerName leSite
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/leSite
  <Directory /var/www/leSite>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>
</VirtualHost>
```

- 2 Modifier `/etc/hosts` en ajoutant :
127.0.0.1 leSite
- 3 Redémarrer Apache : `/etc/init.d/apache2 restart`
- 4 Ajouter si nécessaire leSite dans “no proxy for” (Firefox) (et travailler éventuellement offline)

Un premier fichier PHP : page.php

Exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
  <head>
    <title>Une page PHP</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  </head>

  <body>
    <p> Contenu généré par PHP ci-dessous :</p>
    <?php
      echo "    <p> Nous sommes le " . date('d/M/Y H:m:s') . " </p> \n";
      echo "    <h2> Bonjour à vous </h2>";
    ?>
  </body>
</html>
```

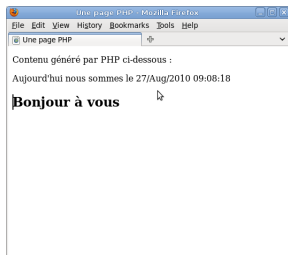
Exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
  <head>
    <title>Une page PHP</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  </head>

  <body>
    <p> Contenu généré par PHP ci-dessous :</p>
    <p> Aujourd'hui nous sommes le 27/Aug/2010 09:08:52 </p>
    <h2> Bonjour à vous </h2>
  </body>
</html>
```

Le résultat à l'écran



Remarque

Dans le code source, les instructions d'affichage ont permis d'assurer la lisibilité du code produit (alignement des balises).

Bloc de code PHP

Un bloc de code (ou script) PHP est une séquence d'instructions PHP se trouvant entre les marqueurs `<?php` et `?>`. Un bloc peut être inclus dans du code HTML ou isolé dans un fichier ne comportant que du code PHP.

L'interpréteur PHP analyse le document, renvoie le code HTML tel quel, mais retourne pour chaque bloc de code PHP une chaîne de caractères (éventuellement vide) qui correspond à l'évaluation des instructions du bloc.

Remarque

Vous pouvez inclure autant de blocs de code PHP que vous le désirez, ce qui veut dire que dans un fichier `.php` vous pouvez à tout moment passer du code PHP au code HTML, et réciproquement.

Warning

Une instruction PHP se termine toujours par un point-virgule.

Écritures alternatives

On peut utiliser une forme courte (short tag) si la directive `short_open_tag` de PHP est activée :

Exemple

```
<? echo "<p> ceci est un autre paragraphe </p>"; ?>
```

Une syntaxe en court-circuit existe qui permet de retourner directement une chaîne de caractères:

Exemple

```
<?="This is another PHP example."; ?>  
// est équivalent à :  
<?php echo "This is another PHP example."; ?>
```

Warning

Les short tags ne sont pas compatibles avec XML (et donc XHTML). Il ne faut donc pas les utiliser.

On peut aussi utiliser la balise `<script>` comme suit :

Exemple

```
<script language="php">  
    echo "<p> ceci est un autre paragraphe </p>";  
</script>
```

Inclusion de fichiers externes

Il est possible d'inclure du code PHP (fonctions, classes) provenant d'autres fichiers avec une directive d'inclusion :

- `include("nomFichier")` remplace cette directive par le contenu intégral du fichier
- `include_once("nomFichier")` garantit l'exécution de la directive une seule fois
- `require("nomFichier")` agit comme la directive `include` mais provoque une erreur fatale en cas de problème
- `require_once("nomFichier")` garantit l'exécution de la directive une seule fois

Exemple

```
<?php
    require_once("header.php");
    ...
    require_once("footer.php");
?>
```

Outline

- 1 Introduction
- 2 Principes de base**
- 3 Les structures de contrôle
- 4 Les chaînes de caractères
- 5 Les tableaux
- 6 Les fonctions
- 7 Modules et fonctions prédéfinis
- 8 Les formulaires
- 9 Cookies et Sessions
- 10 Accès à MySQL

Les commentaires

Les commentaires peuvent être introduits sur une seule ligne à la C++, en préfixant par //, ou sur plusieurs lignes à la C, entre /* et */.

Commentaires à la C++

```
<?php
  // Title: My PHP Program
  // Author: Jason
  ...
?>
```

Commentaires à la C

```
<?php
  /*
     Title: My PHP Program
     Author: Jason
  */
  ...
?>
```

Remarque

On peut également commenter sur une seule ligne à la UNIX en préfixant par #.

Warning

Les commentaires ne sont pas présents dans le code HTML généré par PHP.

Les variables

Une variable est le conteneur d'une valeur pour l'un des types utilisés par PHP.

Une variable :

- possède un identifiant qui commence toujours par le caractère \$
- doit commencer par une lettre (après le \$), ou le caractère _
- est sensible à la casse
- ne réclame ni déclaration ni initialisation obligatoires
- peut recevoir une valeur en utilisant l'opérateur = d'affectation

Exemple

```
$x=10;  
$ville="lille";  
$codePostal=59800;
```

Remarque

Je vous recommande d'utiliser le schéma de nommage `camel case`



La syntaxe de l'affectation (par valeur) est:

```
nomVariable = expression ;
```

L'expression peut inclure des constantes, de opérateurs arithmétiques, logiques, des appels de fonction, ...

L'affectation par référence consiste à créer un alias d'une variable. La nouvelle variable créée et l'ancienne représentent le même conteneur. La syntaxe de l'affectation (par référence) est :

```
nomVariable1 = &nomVariable2 ;
```

Remarque

Tout changement apportée sur une variable est également effectif sur les alias.

Code PHP

```
$ville1="paris";  
echo "ville1=" . $ville1 . "<br />";  
$ville2="Lyon";  
echo "ville2=" . $ville2 . "<br />";  
$ville2=&$ville1;  
echo "ville1=" . $ville1 . " ville2=" . $ville2 . " <br />";  
$ville1="lille";  
echo "ville1=" . $ville1 . " ville2=" . $ville2 . " <br />";
```

Résultat de l'exécution

```
ville1=paris  
ville2=Lyon  
ville1=paris ville2=paris  
ville1=lille ville2=lille
```

Remarque

Le symbole . est l'opérateur de concaténation.

Les types

Dans PHP, il n'existe pas de déclaration explicite du type d'une variable lors de sa création. Toutefois, une variable possède toujours un type (éventuellement NULL), à un instant donné. Les types scalaires sont :

- `boolean` : valeurs `false` et `true`, insensibles à la casse
- `integer` : nombres entiers allant typiquement jusqu'à $+/-2^{31}$ pour PHP 5 et $+/-2^{63}$ pour PHP 6.
- `double` : valeurs réelles
- `string` : chaînes de caractères

Les types spéciaux sont :

- `resource` : type d'une variable qui représente une référence (un identifiant) sur le serveur
- `NULL` : type d'une variable non initialisée, ou initialisée avec la valeur `null`

Remarque

On peut utiliser `0` pour représenter `false` et n'importe quelle valeur différente de `0` pour représenter `true`.

Les types composés sont :

- array : type d'une variable représentant un tableau
- object : type d'une variable représentant un objet.

PHP effectue parfois des conversions implicites. Par exemple:

Exemple

```
$x=15; // x est une variable integer  
$x="paris"; // x est maintenant une variable string
```

Il est possible d'effectuer des conversions explicites (transtypage) avec la syntaxe :
nomVariable = (type) expression;

Par exemple:

Exemple

```
$x=152.12; // x est une variable double  
$y=(integer)$x; // y est une variable integer de valeur 152
```

Remarque

La fonction `settype()` permet de modifier le type d'une variable.

Les opérateurs numériques

Ce sont les opérateurs classiques :

- +, -, *, /
- % (modulo ou reste de la division entière)
- ++ (incrément)
- -- (décrément)

Exemple

```
$x=15;  
echo $x%7; // affiche 1 car 15=7*2+1  
$y=$x++;  
echo $y; // affiche 16
```

Remarque

Les fonctions mathématiques prédéfinies sont décrites ici :

<http://www.php.net/manual/en/ref.math.php>

Les opérateurs d'affectation combinée

On trouve :

- +=, -=, *=, /= et %= comme en C, C++ et Java
- .= concaténation puis affectation

Exemple

```
$x += $y$; // équivaut à $x = $x + $y;  
$s = "bonjour";  
$t = "toto";  
$s .= " " . $t;  
echo $s; // affiche bonjour toto
```

Remarque

L'opérateur .= est assez pratique.

Les opérateurs booléens

On trouve les opérateurs de comparaison classiques :

- `<`, `<=`, `>`, `>=`
- `==` teste l'égalité de 2 valeurs
- `===` teste si 2 valeurs sont identiques (égalité des valeurs et des types)
- `!=` teste l'inégalité de 2 valeurs
- `!==` teste la non-identité de deux valeurs

Exemple

```
$x = 15;  
$y = "15";  
$b = $x==$y; // b vaut true  
$c = $x===$y; // c vaut false
```

Warning

Attention à ne pas confondre `=` avec `==` dans l'écriture d'une comparaison.

On trouve les opérateurs logiques :

- or et || (équivalent à or mais avec une priorité différente)
- and et && (équivalent à and mais avec une priorité différente)
- xor
- !

Exemple

```
$x == true or ($a==$b); // si x vaut true, $a==$b n'est pas évaluée  
$y == false and f($z); // si y vaut false, f n'est pas appelée
```

Exemple

```
mysql_connect("host", "user", "pass") or exit("Unable to connect");
```

Warning

Ces opérateurs sont coupe-circuits : si le premier opérande permet de déterminer le résultat, le second opérande n'est pas évalué.

Fonctions sur les types

La fonction `gettype()` retourne le type d'une variable. Pour déterminer si une variable est d'un type précis, on utilise :

```
is_bool()      is_integer()  is_double()    is_string()
is_array()     is_object()   is_resource()  is_null()
is_numeric()   is_scalar()
```

Exemple

```
$data = array(1, 3.67, null, false, "toto");
foreach ($data as $value)
    echo gettype($value) . " ";
echo "<br />" . gettype($data);
// donne :
integer double NULL boolean string
array
```

Contrôler l'état d'une variable

Deux fonctions permettent de contrôler l'état des variables :

- `isset()` : retourne la valeur `true` ssi la variable passée en paramètre a été initialisée (à une valeur différente de `null`).
- `empty()` : retourne la valeur `true` ssi la variable passée en paramètre a une valeur vide, i.e., `null`, `false`, `0` (0 comme integer), `""` (la chaîne vide), `"0"` (0 comme string), `array()` (un tableau vide), ou une variable déclarée sans valeur dans une classe.

Exemple

```
$b=0;
if (isset($b)) ... // condition évaluée à true
if (empty($b)) ... // condition évaluée à true
if (isset($c)) ... // condition évaluée à false
if (empty($c)) ... // condition évaluée à true
```

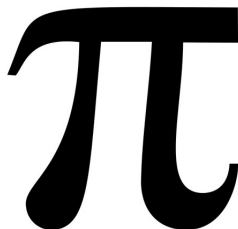
Remarque

Ces fonctions sont particulièrement utiles pour tester (l'existence d') une réponse dans un champ de saisie d'un formulaire.

Pour définir une constante, il faut utiliser la fonction `define()` qui retourne la valeur `true` si la constante a été bien créée. La fonction `defined()` permet de savoir si une constante donnée existe déjà.

Notons que :

- le nom d'une constante ne commence pas par \$
- les constantes sont globales; elles peuvent être utilisées partout dans un script



Exemple

```
define("NB_JOUEURS",10);  
echo "la constante NB_JOUEURS vaut " . NB_JOUEURS . " <br /> ";  
if (defined("PI"))  
    echo "la constante PI est définie <br /> ";
```

Exemple

```
print_r(get_defined_constants());
```

Exemple

```
[PHP_VERSION] => 5.3.2-1ubuntu4.2  
[PHP_MAJOR_VERSION] => 5  
[PHP_OS] => Linux
```

Remarque

Écrivez les constantes en majuscules avec le tiret bas comme séparateur entre les mots.

Outline

- 1 Introduction
- 2 Principes de base
- 3 Les structures de contrôle**
- 4 Les chaînes de caractères
- 5 Les tableaux
- 6 Les fonctions
- 7 Modules et fonctions prédéfinis
- 8 Les formulaires
- 9 Cookies et Sessions
- 10 Accès à MySQL

Structures de contrôle

Elles sont proches de celles de langages tels que C, C++, Java, Javascript. Pour rappel, les structures de contrôles sont de trois types :

- **la séquence** : exécution séquentielle d'une suite d'instructions séparées par un point-virgule
- **l'alternative** : structure permettant un choix entre divers blocs d'instructions suivant le résultat d'un test logique
- **la boucle** : structure itérative permettant de répéter plusieurs fois le même bloc d'instructions tant qu'une condition de sortie n'est pas avérée

Remarque

Toute condition utilisée dans l'une de ces structures sera toujours placée entre parenthèses.

Warning

Il ne faut pas confondre $x == y$ (test d'égalité) avec $x = y$ (affectation).

L'instruction **if** sans partie else :

```
if (condition) instruction;
```

```
if (condition) {  
    instruction;  
}
```

```
if (condition) {  
    instruction1;  
    instruction2;  
    ...  
}
```

Exemple

```
if ($x >= 0) echo "valeur positive ou nulle";  
...  
if ($note > 12 && $note <= 14) {  
    echo "bravo" ;  
    $mention="bien";  
}
```

Alternatives

L'instruction **if...else** :

```
if (condition) instruction1;  
else instruction2;
```

```
if (condition) {  
    instructions1;  
} else {  
    instructions2;  
}
```

```
if (condition1) {  
    instructions1;  
} elseif (condition2) {  
    instructions2;  
} else {  
    instructions3;  
}
```

Exemple

```
if ($rank == 1)  
    $medaille="or";  
elseif ($rank == 2)  
    $medaille="argent";  
elseif ($rank == 3)  
    $medaille="bronze";
```

L'instruction **switch** :

```
switch (expression) {  
    case valeur1 :  
        instructions1;  
        break;  
    case valeur2 :  
        instructions2;  
        break;  
    ...  
    case valeurN :  
        instructionsN;  
        break;  
    default:  
        instructionsDefault;  
}
```

Remarque

Le branchement par défaut n'est pas obligatoire.

L'opérateur ternaire ?: permet de remplacer une instruction if...else simple. Sa syntaxe (lorsqu'utilisée pour donner une valeur à une variable) est :

```
variable = condition ? expressionIf : expressionElse;
```

Elle est équivalente à :

```
if (condition) variable=expressionIf;  
else variable=expressionElse;
```

Exemple

```
$civilite = ($sexe == "F") ? "Madame" : "Monsieur";
```

Remarque

Cet opérateur est utile pour les expressions courtes.

L'instruction **while** :

```
while (condition) instruction;
```

```
while (condition) {  
    instruction1;  
    instruction2;  
    ...  
}
```

Exemple

```
$num = 1;  
while ($num <= 5 ) {  
    echo $num . "<br />";  
    $num++;  
}
```

L'instruction **for** :

```
for (instructionInit; condition; instructionIter) instruction;  
  
for (instructionInit; condition; instructionIter) {  
    instruction1;  
    instruction2;  
    ...  
}
```

Exemple

```
for ($num=1; $num <= 5; $num++)  
    echo $num . "<br />";
```

Remarque

Il n'est pas nécessaire de placer des parenthèses autour de la condition pour le for.

L'instruction **do...while** :

```
do {  
    instruction1;  
    instruction2;  
    ...  
}  
while (condition);
```

L'instruction **foreach** pour les tableaux :

```
foreach ($t as $valeur) {  
    ...  
}  
  
foreach ($t as $cle=>$valeur) {  
    ...  
}
```

Certaines instructions permettent un contrôle supplémentaire sur les boucles :

- **break** permet de quitter la boucle courante
- **break n** permet de quitter les n boucles (les plus internes) courantes
- **continue** permet de terminer l'itération en cours de la boucle courante
- **continue n** permet de terminer l'itération en cours des n boucles (les plus internes) courantes

Exemple

```
for ($i=0; $i<5; $i++) {  
  for ($j=0; $j<5; $j++) {  
    for ($k=0; $k<5; $k++) {  
      if (($i+$j+$k)%3 == 0)  
        continue 3;  
      echo "$i $j $k <br />":  
    }  
  }  
}
```

Exercice

Écrire le code PHP qui détermine si un nombre entier x est parfait.

Definition

Un nombre est parfait ssi il est égal à la somme de ses diviseurs.

6 est parfait car $6 = 1 + 2 + 3$.



Outline

- 1 Introduction
- 2 Principes de base
- 3 Les structures de contrôle
- 4 Les chaînes de caractères**
- 5 Les tableaux
- 6 Les fonctions
- 7 Modules et fonctions prédéfinis
- 8 Les formulaires
- 9 Cookies et Sessions
- 10 Accès à MySQL

Simple ou doubles quotes

Une chaîne de caractères en PHP peut être définie de deux manières :

- entre simple quotes ou apostrophes '
- entre double quotes ou guillemets "

On utilisera l'une ou l'autre notation pour éviter les problèmes de fermeture intempestive.

Exemple

```
echo "Bonjour, je m'appelle Henri";  
echo 'Tom a dit "je vous aime"';
```

Toutefois, il est toujours possible de protéger certains caractères avec le caractère d'échappement (antislash).

Exemple

```
echo 'Bonjour, je m\'appelle Henri';  
echo "Tom a dit \"je vous aime\"";
```

Les séquences d'échappement utiles en PHP pour la notation double quote (pour les simples quotes, seuls ' et \ peuvent être échappés) sont :

Séquence	Signification
\'	affiche une apostrophe
\"	affiche des guillemets
\\$	affiche \$
\\	affiche \
\n	nouvelle ligne
\r	retour chariot
\t	tabulation

Si une chaîne contient une variable, les deux types de notation ne sont plus équivalents. La valeur de la variable sera expansée (incorporée à la chaîne) si vous utilisez des guillemets, contrairement aux apostrophes.

Exemple

```
$nom = "toto";  
echo "Bonjour $nom"; // affiche Bonjour toto  
echo 'Bonjour $nom'; // affiche Bonjour $nom
```

Concaténation

L'opérateur de concaténation est le caractère point(.). Il permet de construire de nouvelles chaînes qui peuvent être affichées ou affectées à de nouvelles variables.

Exemple

```
$s = "Bonjour";  
$nom = "toto";  
echo $s . " " . $nom; // affiche Bonjour toto  
$mess = $s . " " . $nom;  
echo $mess; // affiche Bonjour toto
```

Remarque

Il est toujours possible d'éviter l'intégration de variables entre double quotes. Par exemple :

```
echo "$prenom $nom est un candidat";  
// est équivalent à :  
echo $prenom . " " . $nom . " est un candidat";
```

Caractères d'une chaîne

Pour accéder à un caractère particulier d'une chaîne, il faut utiliser les crochets comme pour un tableau. Le premier caractère se trouve en position 0. Le dernier caractère se trouve à la position juste avant la valeur retournée par `strlen()`.

Exemple

```
$s="monde";  
for ($i=0; $i<strlen($s); $i++)  
    echo $s[$i] . " "; // affiche "m o n d e "  
$s[0]='r';  
echo $s; // affiche "ronde"
```

Pour obtenir le code ASCII d'un caractère, utiliser la fonction `ord()`. Pour obtenir le caractère ASCII d'un code donné, utiliser la fonction `chr()`. Par exemple, pour créer un mot de passe :

Exemple

```
for ($i=0; $i<8; $i++)  
    $pass .= chr(rand(65,90));
```

Modification de la casse et des espaces

Il existe plusieurs fonctions :

- `strtolower()` : retourne une chaîne donnée en minuscule
- `strtoupper()` : retourne une chaîne donnée en majuscule
- `ucwords()` : retourne une chaîne donnée avec les initiales des mots en majuscule
- `ucfirst()` : retourne une chaîne donnée avec la première lettre en majuscule

Exemple

```
$nom="duPont"; $prenom="JEan";  
$adresse="21 rue gambetta"; $ville="paris";  
echo ucfirst(strtolower($prenom)) . " " . strtoupper($nom) . ", ";  
echo ucwords(strtolower($adresse)) . " " . strtoupper($ville);
```

affiche : Jean DUPONT 21, Rue Gambetta PARIS

Remarque

Pour éliminer les espaces (voire d'autres caractères) en début et/ou fin de chaîne, utiliser `ltrim()`, `rtrim()` et `trim()`.

Quelques autres fonctions

Séquence	Signification	
array	<code>explode(string separator, string s)</code>	éclate s
string	<code>implode(string glue, array pieces)</code>	recolle les pièces
int	<code>strlen(string s)</code>	longueur de s
int	<code>strpos(string s, string m)</code>	position du 1er motif dans s
int	<code>strrpos(string s, string m)</code>	position du der motif dans s
string	<code>strrev(string s)</code>	inverse une chaîne
string	<code>substr(string s, int start, int length)</code>	extraît une chaîne
int	<code>strcmp(string s1, string s2)</code>	compare les chaînes

Exemple

```
print_r(explode("/", "/home/toto/php/chaine/essai.php"));  
// affiche Array ( [0] => [1] => home [2] => toto [3] => php  
                  [4] => chaine [5] => essai.php )
```

Pour améliorer la présentation HTML des textes saisis dans des formulaires (dans des `<textarea>` par exemple), il est judicieux de transformer les sauts de ligne `"\n"` non interprétés par HTML en balise `
`. La fonction `n12br()` permet de faire cela automatiquement.

Exemple

```
$s = "salut \n toto \n titi \n";  
echo n12br($s); // affiche "salut <br /> toto <br /> titi <br />"
```

Pour sécuriser l'affichage HTML des textes saisis par l'utilisateur, il faut utiliser la fonction `strip_tags()`. Imaginez par exemple que l'utilisateur insère : `<script type="text/javascript"> alert("coucou"); </script>`

Exemple

```
$text = "<p>Test.</p> <!-- Comment --> <a href=\"#top\">top</a>";  
echo strip_tags($text) . "\n"; // affiche "Test. top"  
echo strip_tags($text, "<p><a>"); // Allow <p> and <a>  
// affiche "<p>Test.</p> <a href=\"#top\">top</a>"
```

Pour protéger votre page tout en préservant les balises, vous pouvez utiliser `htmlspecialchars()`. Cette fonction transforme les caractères qui ont une signification spéciale en HTML en leurs entités de caractère :

- & (ampersand) devient `&`;
- " (double quote) devient `"`; si `ENT_QUOTES` n'est pas activée
- ' (single quote) devient `'`; lorsque `ENT_QUOTES` est activée
- < (less than) devient `<`; > (greater than) devient `>`;

Exemple

```
$s = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);  
echo $s; // affiche &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;
```

Remarques

Un deuxième paramètre peut être utilisé pour déterminer quelles quotes échapper exactement.

La fonction `htmlentities()` va plus loin que `htmlspecialchars()` en remplaçant tous les caractères dont le code UNICODE est supérieur à 128.

Certains caractères (simple et double quotes, \, le caractère NUL) présents dans les chaînes devant être enregistrées dans une base de données posent problème. La fonction addslashes() permet de les échapper.

Exemple

```
$s = addslashes("c'est c:\system\php5");  
echo $s; // affiche "c\'est c:\\system\\php5"  
echo stripslashes($s); // affiche "c'est c:\system\php5"
```

Remarque

La fonction quotemeta() va plus loin en échappant d'autres (méta-)caractères.

Warning

Les fonctions addslashes() et stripslashes() sont inutiles si la directive magic_quotes_gpc (et magic_quotes_runtime) est active dans php.ini. gpc fait référence à get, post et cookie.

Outline

- 1 Introduction
- 2 Principes de base
- 3 Les structures de contrôle
- 4 Les chaînes de caractères
- 5 Les tableaux**
- 6 Les fonctions
- 7 Modules et fonctions prédéfinis
- 8 Les formulaires
- 9 Cookies et Sessions
- 10 Accès à MySQL

Les tableaux indicés

Pour ce type de tableaux, on utilise une valeur numérique pour accéder à une case du tableau.

Exemple

```
$t = array(10, 100, 1000);
print_r($t); // Array ( [0] => 10 [1] => 100 [2] => 1000 )
echo $t[1]; // 100
echo $t[100]; // rien d'affiché
$t[2]=12;
print_r($t); // Array ( [0] => 10 [1] => 100 [2] => 12 )
$t2 = array("banane","orange");
print_r($t2); // Array ( [0] => banane [1] => orange )
echo $t2[0]; // banane
$t3[3]=5;
$t3[4]="hello";
$t3[]=8;
print_r($t3); // Array ( [3] => 5 [4] => hello [5] => 8 )
echo count($t) . " " . count($t2) . " " . count($t3); // 3 2 3
```

Les tableaux associatifs

Pour ce type de tableaux, on utilise une chaîne de caractères (appelée clé) pour accéder à une case du tableau.

Exemple

```
$t['banane'] = 5; $t['orange'] = 10; $t['pomme'] = 3;
print_r($t); // Array ( [banane]=>5 [orange]=>10 [pomme]=>3 )
echo $t['orange']; // 10
echo $t['poire']; // rien d'affiché
echo $t[100]; // rien d'affiché
$t['banane']=12;
print_r($t); // Array ( [banane]=>12 [orange]=>10 [pomme]=>3 )
echo count($t); // 3
```

Warning

L'utilisation d'éléments de tableaux associatifs dans des chaînes pose problème :

```
echo "la valeur est $t['pomme']"; n'est pas valide.
echo "la valeur est {$t['pomme']}"; est valide.
echo "la valeur est" . $t['pomme']; est valide.
```

Création de tableaux

On peut créer un tableau en plaçant une première valeur par simple affectation, comme par exemple `$t[3]=10;`. Pour déclarer un tableau avec la possibilité d'inclure de suite plusieurs valeurs, on utilisera la fonction `array()`.

Exemple

```
$t = array(10,100,1000);  
$ta = array('banane'=>5, 'orange'=>10, 'pomme'=>3);  
$m = array(array(1,2,3), array(4,5,6));  
$m[1][2]=66;  
var_dump($m); // array(2) {  
  [0]=> array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }  
  [1]=> array(3) { [0]=> int(4) [1]=> int(5) [2]=> int(66) } }
```

Remarque

On peut définir des tableaux à autant de dimensions que désiré.

L'instruction foreach

Pour tout tableau, on peut parcourir ses éléments de deux manières différentes avec foreach :

```
foreach ($tableau as $valeur) ...
```

ou

```
foreach ($tableau as $cle=>$valeur) ...
```

Exemple

```
$t = range(5,8);  
foreach ($t as $valeur)  
    echo $valeur . " "; // 5 6 7 8  
foreach ($t as $cle=>$valeur)  
    echo "($cle,$valeur) "; // (0,5) (1,6) (2,7) (3,8)  
$ta = array('banane'=>5, 'orange'=>10, 'pomme'=>3);  
foreach ($ta as $valeur)  
    echo $valeur . " "; // 5 10 3  
foreach ($ta as $cle=>$valeur)  
    echo "($cle,$valeur) "; // (banane,5) (orange,10) (pomme,3)
```

Ajouter et supprimer des éléments

On peut utiliser les fonctions suivantes :

- `array_push()` insère en fin de tableau
- `array_unshift()` insère en début de tableau
- `array_pop()` supprime le dernier élément
- `array_shift()` supprime le premier élément
- `unset()` supprime un élément donné

Exemple

```
$t = range(5,8);  
array_pop($t);  
array_push($t,10);  
array_unshift($t,9);  
print_r($t); // Array ( [0]=>9 [1]=>5 [2]=>6 [3]=>7 [4]=>10 )  
$ta = array('banane'=>5, 'orange'=>10, 'pomme'=>3);  
array_shift($ta);  
unset($ta['pomme']);  
print_r($ta); // Array ( [orange]=>10 )
```

Fonctions diverses sur les tableaux

Quelques fonctions pour agir sur les tableaux.

- `array_unique()` retourne un tableau ne comportant que la dernière occurrence de chaque valeur d'un tableau
- `array_merge()` retourne un tableau fusion de deux autres tableaux
- `array_intersect()` retourne un tableau intersection de deux autres tableaux
- `array_diff()` différence d'un premier tableau par un second tableau
- `array_reverse()` inverse l'ordre des éléments
- `shuffle()` mélange de manière aléatoire les éléments d'un tableau

Quelques fonctions de tri :

- `sort()` et `asort()` tri par ordre croissant
- `rsort()` et `arsort()` tri par ordre décroissant
- `natsort()` tri selon l'ordre naturel
- `usort()` et `uasort()` tri utilisateur en définissant une fonction de comparaison
- `krsort()`, `ksort()` et `uksort()` tri selon les clés d'un tableau associatif

On dispose d'un tableau associatif `$resultats` dont chaque clé correspond à une équipe de football et chaque valeur à une lettre parmi D (pour défaite), N (pour nul) et V (pour victoire). Écrire le code PHP qui permet de compter le nombre d'équipes ayant gagné.



Outline

- 1 Introduction
- 2 Principes de base
- 3 Les structures de contrôle
- 4 Les chaînes de caractères
- 5 Les tableaux
- 6 Les fonctions**
- 7 Modules et fonctions prédéfinis
- 8 Les formulaires
- 9 Cookies et Sessions
- 10 Accès à MySQL

Fonctions sans paramètres

La forme est la suivante :

```
function nomFonction() {  
    instructions;  
}
```

Exemple

```
<head>  
  <?php  
    function bienvenue() {  
      echo "<h2> Bonjour, bienvenue sur mon site </h2>";  
      echo "<p> Nous sommes le " . date("d/m/Y") . "<br /> </p>";  
    }  
  ?>  
</head>  
<body>  
  <?php bienvenue(); ?>  
  <p> A vous de jouer </p>  
</body>
```

Fonctions avec paramètres

La forme est la suivante :

```
function nomFonction($param1, $param2, ..., $paramN) {  
    instructions;  
}
```

Exemple

```
function maxDe($val1, $val2) {  
    if ($val1 > $val2) echo $val1;  
    else echo $val2;  
}  
...  
echo "Valeur max = ";  
maxDe($v1,$v2);
```

Remarque

Il n'est pas possible de redéfinir une fonction existante (ici, la fonction ne pouvait pas s'appeler max() car elle existe déjà).

Fonctions avec retour

Une fonction peut retourner une valeur avec l'instruction `return` :

Exemple

```
function maxDe($val1, $val2) {  
    if ($val1 > $val2)  
        return $val1;  
    else  
        return $val2;  
}  
...  
echo "Valeur max = " . maxDe($v1,$v2);
```

Remarque

Il est possible de retourner plusieurs valeurs en utilisant un tableau (par exemple associatif).

Paramètres par défaut

Cela consiste à placer une valeur au moment de la définition du paramètre :

Exemple

```
function ht($prix, $taux=19.6) {  
    return round($prix*(1-$taux/100),2);  
}  
...  
echo "Prix HT = " . ht(10000,5.5);  
echo "Prix HT = " . ht($montant);
```

Warning

Tous les paramètres qui ont une valeur par défaut doivent figurer en dernier lors de la définition.

Nombre variable de paramètres

Vous pouvez donner plus de paramètres effectifs (paramètres apparaissant lors de l'appel) que de paramètres formels (paramètres apparaissant lors de la définition).

Vous pouvez alors utiliser les fonctions :

- `func_num_args()` retourne le nombre de paramètres effectivement passés
- `func_get_args()` retourne un tableau avec tous les paramètres effectifs
- `func_get_arg()` retourne le *i*ème paramètre (à partir de 0)

Exemple

```
function bienvenue() {  
    $nom = func_num_args() == 1 ? " " . func_get_arg(0) : "";  
    echo "<h2> Bonjour$nom, bienvenue sur mon site </h2>";  
}  
...  
bienvenue(); bienvenue("toto");
```

Warning

Cette technique permet de pallier à l'impossibilité de surcharger les fonctions.

Passage de paramètres par référence

Les paramètres sont passés par valeur par défaut. Toute modification de la valeur d'un paramètre n'a aucune répercussion sur la valeur du paramètre effectif correspondant. Pour utiliser la passage de paramètre par référence, on place le symbole & devant le nom du paramètre formel.

Exemple

```
function swap(&$val1, &$val2) {  
    $tmp = $val1;  
    $val1 = $val2;  
    $val2 = $tmp;  
}  
...  
$x=10; $y=20;  
swap($x,$y);  
echo "$x $y"; // affiche 20 10
```

Remarque

Il existe également un passage occasionnel de paramètre par référence (déprécié).

Fonctions dynamiques

Le nom d'une fonction connue seulement à l'exécution peut être appelée en plaçant celui-ci dans une variable et en faisant suivre (pour un appel) cette variable des parenthèses et des paramètres éventuels.

Exemple

```
$s = "date";  
$s("d/M/Y"); // affiche 29/08/2010  
$t = array("min","max");  
foreach ($t as $f)  
    echo "$f=" . $f(10,20) . " "; // affiche min=10 max=20
```

Remarque

Cette technique est liée à la “reflection/introspection” que l'on trouve dans certains langages tels que Java.

Variables locales vs variables globales

Une **variable globale** est une variable définie dans un script en dehors de toute fonction et classe. Une **variable locale** est toute variable utilisée dans une fonction, sauf si il y a indication contraire.

Exemple

```
$interne="dedans";
$externe="dehors";
function waytu($interne) {
    $interne="je suis $interne";
    $externe="n'importe quoi!";
    return $interne;
}
echo waytu($interne); // affiche je suis dedans
echo $interne; // affiche dedans
echo waytu($externe); // affiche je suis dehors
echo $externe; // affiche dehors
```

Variables locales vs variables globales

Pour utiliser une variable globale au sein d'une fonction il faut la déclarer en la faisant précéder du mot-clé `global`, ou en y accédant directement avec le tableau associatif `GLOBALS`.

Exemple

```
function sum() {  
    global $a, $b;  
    $b = $a + $b;  
}  
$a = 1;  
$b = 2;  
sum();  
echo $b; // affiche 3
```

Exemple

```
function sum() {  
    $GLOBALS['b'] = $GLOBALS['a'] +  
        $GLOBALS['b'];  
}  
$a = 1;  
$b = 2;  
sum();  
echo $b; // affiche 3
```

Remarque

Les variables globales sont à éviter dans les fonctions (en particulier, si c'est pour les modifier) dans la mesure du possible.

Elles apparaissent dans des tableaux dits superglobaux :

- `$_ENV` contient le nom et la valeur des variables d'environnement
- `$_SERVER` contient les informations liées au serveur web
- `$_GET` contient le nom et la valeur des données issues d'un formulaire envoyé par la méthode get
- `$_POST` contient le nom et la valeur des données issues d'un formulaire envoyé par la méthode post
- `$_FILES` contient le nom des fichiers téléchargés à partir du poste client
- `$_COOKIE` contient le nom et la valeur des cookies enregistrées sur le poste du client
- `$_SESSION` contient l'ensemble des noms de variables de session et leurs valeurs

Warning

Pour des raisons de sécurité, éviter d'utiliser les tableaux superglobaux `$GLOBALS` et `$_REQUEST`, non listés ci-dessus.

Variables statiques

Pour conserver la valeur précédemment affectée à une variable locale d'une fonction entre deux appels, il faut la déclarer statique en la faisant précéder du mot-clé `static`. Ces variables permettent d'effectuer des opérations de cumul.

Exemple

```
function compterAppels() {  
    static $nbAppels = 0;  
    $nbAppels++;  
}
```

Remarques

Il faut déclarer la variable avant toute utilisation.

Une variable statique ne conserve une valeur que pendant la durée du script.

Outline

- 1 Introduction
- 2 Principes de base
- 3 Les structures de contrôle
- 4 Les chaînes de caractères
- 5 Les tableaux
- 6 Les fonctions
- 7 Modules et fonctions prédéfinis**
- 8 Les formulaires
- 9 Cookies et Sessions
- 10 Accès à MySQL

Modules et fonctions PHP

Pour vérifier la disponibilité des modules PHP sur votre serveur, utiliser la fonction `get_loaded_extensions()` qui retourne un tableau contenant les modules installés. Pour un module donné, la fonction `get_extension_funcs()` retourne les fonctions disponibles.

Exemple

```
$modules = get_loaded_extensions();
foreach ($modules as $module) {
    echo "<h2>" . $module . "</h2>";
    $functions = get_extension_funcs($module);
    foreach ($functions as $function)
        echo " " . $function;
}
if (function_exists("mail"))
    echo "mail est disponible";
```

Remarque

La fonction `function_exists()` teste la disponibilité d'une fonction donnée.

Si la fonction mail est disponible et un serveur SMTP disponible, vous pouvez effectuer un envoi automatique d'e-mails.

Exemple

```
$name=$_POST['txtName'];
$from=$_POST['txtMail'];
$feedback=$_POST['txtFeed'];

$to = "toto@serveur.com";
$subject = "Feedback from web site";
$content = "Name: " . $name . "\n" . "Email: " . $from . "\n"
           . "Comments:\n" . $feedback . "\n";
$headers = "From: " . $from . "\r\n"
           . "X-Mailer: PHP/" . phpversion();

function_exists(mail) or exit("mail non dispo");
if (mail($to, $subject, $content, $headers))
    echo "mail envoyé";
else echo "mail non envoyé";
```

Timestamps

Les informaticiens ont défini une date d'origine arbitraire, correspondant au 1er Janvier 1970 00h 00m 00s. A partir de cette date, le temps est compté en secondes. Ce nombre de secondes est appelé timestamp ou instant UNIX.

Quelques fonctions PHP :

- `time` retourne le timestamp courant
- `mktime` retourne un timestamp correspondant à une date donnée
- `checkdate` contrôle la validité d'une date

Remarque

Un timestamp permet de stocker facilement une date dans une base de données.

Warning

Les timestamps négatifs ne sont pas gérés par les fonctions de date PHP sous Windows.

Afficher une date

La fonction `date()` permet d'afficher une date en utilisant une chaîne de caractères spéciale de formatage. La fonction `getdate()` retourne un tableau contenant toutes les informations de base :

Clé	Signification
<code>wday</code>	le jour de la semaine de 0 à 6
<code>weekday</code>	le jour de la semaine en lettres
<code>mday</code>	le jour du mois de 1 à 31
<code>month</code>	le mois de 1 à 12
<code>year</code>	l'année sur 4 chiffres
<code>hours</code>	l'heure de 0 à 23
<code>minutes</code>	les minutes de 0 à 59
<code>seconds</code>	les secondes de 0 à 59

Exemple

```
$d = getdate();  
echo "Nous sommes" . $d['weekday'] . " " . $d['mday'] . " "  
    . $d['month'] . " " . $d['year'];
```

Il est possible de créer des images dynamiquement en utilisant l'extension GD (Graphic Device). On utilise les fonctions :

- `imagecreate` pour créer (le cadre de) l'image
- `imagecolorallocate` pour définir une couleur
- `imagepng` pour enregistrer l'image sur le serveur et/ou l'envoyer au client
- `imagesetpixel` pour colorier un pixel
- `imageline`, `imagerectangle`, `imagepolygon`, `imagearc`, `imageellipse`
- `imagefilledrectangle`, `imagefilledpolygon`, `imagefilledarc`, `imagefilledellipse` : tracés pleins
- `imagefill` pour remplir une surface avec une couleur
- `imagesettile` pour remplir une surface avec un motif
- `imagestring` pour afficher un texte

Exemple

```
function histo($x,$y,$data,$texte,$titre) {
    $image=imagecreate($x,$y);
    $ocre=imagecolorallocate($image,195,155,125); // pour le fond
    $blanc=imagecolorallocate($image,255,255,255);
    ...
    for ($i=0; $i<count($data); $i++) {
        //tracés des rectangles en noir
        imagerectangle($image,$cx[$i],$cy[$i],$cx[$i],$y0,$noir);
        //remplissage des rectangles en jaune
        imagefill($image,$cx[$i]+5,$y0-5,$jaune);
        //Ecriture de texte au dessus des rectangles
        imagestring($image,4,$cx[$i],$cy[$i], $data[$i],$noir);
    }
    //enregistrer l'image
    imagepng($image,"images/histo.png");
    //Libérer la mémoire
    imagedestroy($image);
}
```

Exemple

```
<div>
  <h2>Résultats des ventes de la semaine</h2>
  <?php
    include("histo.php");
    $data = array(850,2500,4050,2730,2075,2590,1450);
    $texte = array("Lun", "Mar", "Mer", "Jeu", "Ven", "Sam", "Dim");
    $titre = "Ventes hebdomadaires PHP 5";
    histo(700,450,$data,$texte,$titre);
  ?>
  
</div>
```

Outline

- 1 Introduction
- 2 Principes de base
- 3 Les structures de contrôle
- 4 Les chaînes de caractères
- 5 Les tableaux
- 6 Les fonctions
- 7 Modules et fonctions prédéfinis
- 8 Les formulaires**
- 9 Cookies et Sessions
- 10 Accès à MySQL

Récupérer les données d'un formulaire

L'attribut `action` de l'élément `<form>` est obligatoire. Il indique où doivent être envoyées les données d'un formulaire. Les différentes possibilités sont :

- une URL absolue (ou relative), comme par exemple :
`action="http://www.example.com/test.php"`
- la désignation du fichier PHP courant à l'aide de l'instruction :
`action="<?= $_SERVER['PHP_SELF'] ?>"`
- la désignation d'une adresse de courrier électronique

Les données sont récupérées par le fichier cible (de l'attribut `action`) grâce :

- au tableau superglobal `$_GET` si l'attribut `method` a la valeur `get`
- au tableau superglobal `$_POST` si l'attribut `method` a la valeur `post`
- au tableau superglobal `$_FILES` si des fichiers sont transférés



Exemple

```
<form action="<?= $_SERVER['PHP_SELF'] ?>" method="post">
  ... <input type="text" name="txtName" id="idName" /> ...
  ... <input type="text" name="txtMail" id="idMail" /> ...
  ... <textarea name="txtFeed" rows="10" cols="60" >
    </textarea>
  <p> <input type="submit" value="Send" /> </p>
</form>
```

```
<?php
$name=$_POST['txtName'];
$email=$_POST['txtMail'];
$feedback=trim($_POST['txtFeed']);

if (!empty($name) && !empty($email) && !empty($feedback))
  echo "<h2>" . htmlspecialchars($name) . ", votre envoi
    a été pris en compte </h2>";
?>
```

Exemple

```
<form action="<?= $_SERVER['PHP_SELF'] ?>" method="get">
  ... <input type="text" name="txtName" id="idName" /> ...
  ... <input type="text" name="txtMail" id="idMail" /> ...
  ... <textarea name="txtFeed" rows="10" cols="60" >
    </textarea>
  <p> <input type="submit" value="Send" /> </p>
</form>
```

```
<?php
$name=$_GET['txtName'];
$email=$_GET['txtMail'];
$feedback=trim($_GET['txtFeed']);

if (!empty($name) && !empty($email) && !empty($feedback))
  echo "<h2>" . htmlspecialchars($name) . ", votre envoi
    a été pris en compte </h2>";
?>
```

Lorsque le script contenant le formulaire est chargé du traitement des données, l'ensemble de la page est réaffichée après traitement, de même que l'ensemble du formulaire. Toutes les saisies sont alors effacées. Pour éviter cela, il faut utiliser les tableaux superglobaux.

Exemple

```
... onclick=""  
  
value="<?= isset($_GET['txtName']) ? $_GET['txtName'] : '' ?>"  
... value="<?= $_GET['txtMail'] ?>" ...  
... <?= $_GET['txtFeed'] ?>
```

Lorsque plusieurs composants possèdent le même nom (par exemple un ensemble de cases à cocher), il est nécessaire faire suivre ce nom de crochets (comme pour créer un tableau).

Exemple

```
XHTML <input type="checkbox" name="competences[]" value="XHTML" />  
PHP <input type="checkbox" name="competences[]" value="PHP" />
```

...

```
$competences=$_POST['competences'];  
echo "Vous avez des compétences informatiques en :";  
foreach($competences as $valeur)  
    echo "$valeur";
```

Charger une autre page

Utiliser du code Javascript avec l'instruction `window.history.back()` ou utiliser la fonction PHP `header` avec une chaîne de caractères commençant par "Location:".

Exemple

```
if (isset($_POST['ident']) && isset($_POST['lang'])) {  
    ...  
} else {  
    echo "<script type='text/javascript'>";  
    echo "alert('Complétez tous les champs');";  
    echo "window.history.back();";  
    echo "</script>";  
}
```

Exemple

```
if(isset($_POST['ident']) && isset($_POST['lang'])) {  
    ...  
} else  
    header("Location:multiple.html");
```

Rappelons que l'élément `<form>` doit être tel que :

- `method="post"`
- `enctype="multipart/form-data"`

Dans le fichier `php.ini`, vous pouvez régler les directives :

- `upload_max_filesize`
- `upload_tmp_dir`

Pour un élément `<input type="file" name="fich" ...`, on obtient un tableau associatif avec :

- `$_FILES['fich']['name']` : le nom du fichier sur le poste client
- `$_FILES['fich']['type']` : le type MIME initial du fichier
- `$_FILES['fich']['size']` : la taille en octets du fichier
- `$_FILES['fich']['tmp_name']` : le nom temporaire du fichier sur le serveur
- `$_FILES['fich']['error']` : le code d'erreur (0 si pas d'erreur)

Le fichier temporaire est perdu lors de la déconnexion. Pour l'enregistrer, il faut utiliser la fonction `move_uploaded_file()`.

Exemple

```
<input type="file" name="fich[]" accept="image/gif"/>
<input type="file" name="fich[]" accept="image/gif"/>
...
if (!empty($_FILES)) {
    $r1=move_uploaded_file($_FILES['fich']['tmp_name'][0], "f1.gif");
    $r2=move_uploaded_file($_FILES['fich']['tmp_name'][1], "f2.gif");
    if ($r1 && $r2) echo "Transferts réalisés !";
    else echo "Transferts non effectués";
}
```

Outline

- 1 Introduction
- 2 Principes de base
- 3 Les structures de contrôle
- 4 Les chaînes de caractères
- 5 Les tableaux
- 6 Les fonctions
- 7 Modules et fonctions prédéfinis
- 8 Les formulaires
- 9 Cookies et Sessions**
- 10 Accès à MySQL

Les cookies

Ce sont de petits fichiers écrits par un script PHP ou d'autres langages sur l'ordinateur du client.

Les cookies sont limités au stockage d'information de petite taille comme le nom, le code d'accès ou les préférences de l'utilisateur.



Remarques

Un site donné ne peut écrire que 20 cookies sur un poste client.
Chaque cookie ne peut dépasser 4Ko.

Warning

Il n'y a aucune garantie d'avoir la possibilité d'enregistrer des cookies chez un client.

Écriture de cookies

Pour écrire un cookie, il faut utiliser la fonction `setcookie`.

Exemple

```
setcookie("nom","toto"); // valable uniquement pour la session  
setcookie("vote","rouge",time()+86400); // valable 24h
```

Pour effacer le contenu d'un cookie, il faut simplement donner le nom du cookie.

Exemple

```
setcookie("nom"); // efface le contenu du cookie "nom"
```

Pour détruire le cookie, il faut définir une date de validité antérieure.

Exemple

```
setcookie("vote","rouge",time()-3600);
```

Warning

Aucun contenu HTML (même un espace) ne doit être envoyé au client avant l'écriture d'un cookie.

Les données stockées dans les cookies pour un site sont récupérables dès le chargement d'une nouvelle page (ou la même) du site. Cela procure un moyen de passage d'information d'une page à l'autre.

Pour récupérer (lire) les cookies, il faut utiliser le tableau superglobal `$_COOKIE` en utilisant comme clé de l'élément recherché le nom du cookie.

Exemple

```
setcookie("achat[0]","livre", time()+3600);
setcookie("achat[1]","CD", time()+3600);
setcookie("achat[2]","bougie", time()+3600);
...
foreach ($_COOKIE['achat'] as $nom=>$valeur)
    echo "Le cookie $nom a pour valeur $valeur <br />";
```

Remarque

Il est possible de construire des tableaux de cookies comme dans l'exemple ci-dessus.

Le mécanisme de sessions permet de gérer des données sur plusieurs pages d'un site sans nécessairement utiliser des cookies. Les étapes sont:

- déclaration (ouverture) de session dans chaque page concernée du site par la fonction `session_start()`
- le client se voit attribuer un identifiant de session à l'ouverture, soit écrit dans un cookie, soit transmis via les URLs
- utilisation du tableau superglobal `$_SESSION` pour gérer les données partagées
- fermeture de la session, avec `session_unset` pour éliminer toutes les variables de session et `session_destroy` pour détruire la session.

Remarque

Les données globales de session (i.e., celles du tableau `$_SESSION`) sont stockées sur le serveur (dans `/tmp` ou un répertoire `session` propre) sous forme de fichiers nommés `sess_` suivi de l'identifiant de session.

Il suffit simplement que l'option adéquate soit activée (voir plus-bas) et d'utiliser `session_start()` comme première instruction, car le cookie est géré automatiquement par PHP.

Exemple

```
<?php // premier fichier page1.php
    session_start();
    $_SESSION['nom']="toto";
    <a href='page2.php'> Vers la page 2 </a>;
...
<?php // second fichier page2.php
    session_start();
    echo "Votre nom est " . $_SESSION['nom'];
...

```

Warning

La directive `session.use_cookies` du fichier `php.ini` doit avoir la valeur `on`, et le poste client doit accepter les cookies.



Il faut ajouter `?<?php echo SID; ?>` à chaque lien.

Exemple

```
<?php // premier fichier page1.php
    session_start();
    $_SESSION['nom']="toto";
    <a href='page2.php?<?php echo SID; ?>'> Vers la page 2 </a>;
...
<?php // second fichier page2.php
    session_start();
    echo "Votre nom est " . $_SESSION['nom'];
...

```

Outline

- 1 Introduction
- 2 Principes de base
- 3 Les structures de contrôle
- 4 Les chaînes de caractères
- 5 Les tableaux
- 6 Les fonctions
- 7 Modules et fonctions prédéfinis
- 8 Les formulaires
- 9 Cookies et Sessions
- 10 Accès à MySQL**

Pour se connecter, il faut indiquer le serveur, le nom d'utilisateur et le mot de passe. On peut définir des constantes dans un fichier `infoConnection.php`.

Exemple

```
<?php
    define("HOST","localhost");
    define("USER","root");
    define("PASS","root");
?>
```

On les utilise comme suit :

Exemple

```
require("infoConnection.php");
$ct=mysql_connect(HOST,USER,PASS) or die("Unable to connect");
mysql_select_db("magasin",$ct) or die("Unable to select the db");
...
mysql_close($ct);
```

La fonction `mysql_connect` retourne une variable de type `resource` qui est un identifiant de connexion utile pour les opérations suivantes.

La fonction `mysql_select_db` précise la base de données à utiliser pour la connexion sur le serveur.

Warning

Il est recommandé de clore la connexion dès que possible en utilisant la fonction `mysql_close`.

Remarque

Le serveur met fin à toute connexion après un délai défini par défaut dans le fichier `php.ini`. La fonction `mysql_ping` permet de vérifier que la connexion est toujours active et effectue une reconnexion automatique le cas échéant.

Pour exécuter une requête, il faut utiliser la fonction `mysql_query`. Le premier paramètre est une chaîne de caractères correspondant à une requête SQL. Le second paramètre est un identifiant de connexion.

Exemple

```
$query="SELECT * FROM article ORDER BY categorie";  
$result=mysql_query($query,$ct) or die("Unable to execute $query");
```

Remarque

Un identifiant de résultat (de type `resource`) est retourné par la fonction `mysql_query`.

Lire les données d'une requête

Après une requête de type SELECT, on dispose d'un identifiant de résultat. La fonction `mysql_fetch_array` permet d'itérer (passer en revue) chaque ligne de la table résultat. Chaque appel retourne un tableau comportant les éléments de la ligne courante, ou `false` si il n'y a plus de ligne.

Exemple

```
while ($row=mysql_fetch_array($result)) {  
    foreach ($row as $key=>$value)  
        echo "$key = $value ";  
    echo "<br />";  
}
```

Remarque

La fonction `mysql_free_result` permet de libérer la mémoire occupée par un identifiant de résultat, et peut être utilisé lorsque vous n'en avez plus besoin.

Pour utiliser le tableau sous forme indicée, on écrira :

Exemple

```
$nbFields= mysql_num_fields($result);
while ($row=mysql_fetch_array($result)) {
    for ($i=0; $i<$nbFields; $i++)
        echo $row[$i] . " ";
    echo "<br />";
}
```

Pour utiliser le tableau sous forme associative, on écrira :

Exemple

```
while ($row=mysql_fetch_array($result)) {
    echo $row['id_article'] . " " . $row['designation'] . " "
        . $row['prix'] . " " . $row['categorie'];
    echo "<br />";
}
```

Il est possible de ne gérer qu'un tableau indicé ou associatif avec :

- `mysql_fetch_row` qui retourne uniquement un tableau indicé ; équivalent à appeler `mysql_fetch_array` avec un deuxième paramètre donné par la constante `MYSQL_NUM`
- `mysql_fetch_assoc` qui retourne uniquement un tableau associatif ; équivalent à appeler `mysql_fetch_array` avec un deuxième paramètre donné par la constante `MYSQL_ASSOC`

Le code des deux exemples de la diapositive précédente reste valable en remplaçant pour le premier `mysql_fetch_array` par `mysql_fetch_row`, et pour le second `mysql_fetch_array` par `mysql_fetch_assoc`.

Quelques fonctions utiles :

- `mysql_num_fields` permet de connaître le nombre de champs (colonnes)
- `mysql_num_rows` permet de connaître le nombre de lignes
- `mysql_field_name` permet de connaître le nom du ième champ
- `mysql_error` retourne le texte d'erreur de la dernière requête (une chaîne vide si il n'y a pas d'erreur)

Exemple

```
$nbFields= mysql_num_fields($result);
$nbRows= mysql_num_rows($result);
for ($i=0; $i<$nbFields; $i++)
    echo mysql_field_name($result,$i) . " ";
for ($i=0; $i<$nbRows; $i++) {
    $row=mysql_fetch_row($result);
    for ($j=0; $j<$nbFields; $j++)
        echo $row[$j] . " ";
}
```

On utilisera une requête SQL avec INSERT. Il faut penser à échapper les caractères qui peuvent poser problème avec addslashes ou plus spécifiquement pour MySQL avec mysql_real_escape_string. Pour placer la valeur NULL, il faut utiliser "\N" (en particulier pour les champs en auto-incrémentation).

Exemple

```
$id_client="\N";
$nom=mysql_real_escape_string($_POST['nom']);
$prenom=mysql_real_escape_string($_POST['prenom']);
...
$query="INSERT INTO client VALUES('$id_client','$nom','$prenom',
    '$age','$adresse','$ville','$mail')";
$result=mysql_query($query,$connection);
if (!$result)
    echo "<script type='text/javascript'> alert('Erreur') </script>";
else
    echo "<script type='text/javascript'> alert('Vous êtes
        enregistré avec le numéro" . mysql_insert_id() . "')</script>";
```



On utilisera une requête SQL avec UPDATE.

Exemple

```
$query = "UPDATE client SET nom='$nom',adresse='$adresse',
  ville='$ville',mail='$mail' WHERE id_client='$code'";
$result=mysql_query($query,$connection);
mysql_close($connection);
if ($result)
  echo "<script type='text/javascript'> alert('Vos modifications
    sont enregistrées'); window.location='index.html';</script>";
```

Remarque

La fonction `mysql_affected_rows` permet de connaître le nombre de lignes affectées par la dernière requête. Cela peut être utile pour contrôler qu'une seule ligne a été modifiée par exemple.

On utilisera une requête SQL avec SELECT.

Exemple

```
$motcle=mysql_real_escape_string($_POST['motcle']);  
$cat=mysql_real_escape_string($_POST['categorie']);  
...  
$queryPart = ($cat == "Tous" ? "" : "AND categorie='$cat'");  
$query = "SELECT id_article AS 'Code article',  
         designation AS 'Description',  
         prix,categorie AS 'Catégorie' FROM article  
WHERE designation LIKE '%$motcle%' $queryPart  
ORDER BY $tri $ordre";
```